Vol.2018-CSEC-83 No.13
2018/12/14

IoT向け属性ベースグループ鍵共有プロトコルの実装

椿 雄介[1,a)]　中西 透[1,b)]

概要：IoT システムでの相互認証・鍵共有において, ID 管理に負担を要する問題を解決するために, 属性ベース暗号を用いた方式が提案・実装されている. しかしこの方式では二者間の通信を想定しており, 複数者間での鍵共有ができない. そこで本研究では, 属性ベース認証付グループ鍵共有プロトコルを IoT を想定した環境で実装する. そしてその性能を PC および Raspberry Pi で評価し, 測定結果から実装システムが実用的であることを確認する.

キーワード：IoT，グループ間鍵共有，属性ベース暗号

An Implementation of Attribute-based Group Key Exchange for IoT

Yusuke Tsubaki[1,a)]　Toru Nakanishi[1,b)]

1. はじめに

近年, IoT を利用したサービスが普及している. IoT システムでは, 機器とサーバ間で双方向の通信モデルとなっているため, 機器とサーバが相互に正当な相手かどうかを検証する相互認証および鍵共有が必須となる. IoT システムにおける相互認証技術の一つに TLS を用いた方式が挙げられるが, TLS は公開鍵証明書を用いるため, システムにおける機器数が多い IoT 環境では証明書利用による処理・通信量の増大や管理の複雑性が課題となる. そのため TLS に代わる新たな方式として, ID ベース暗号を用いた TLS が提案されている [2]. ID を公開鍵として利用する ID ベース暗号では公開鍵証明書が不要であるため, 証明書の検証や送受信の処理を省くことができる. しかし, ID ベース暗号では機器とそれぞれに ID が付与され, その個別の ID を用いて認証を行うため, 機器数が非常に多くなる IoT 環境では ID 管理に負担を要する. そこで復号者を属性の条件によってまとめて指定可能な暗号方式である属性ベース暗号 [4] を用いることにより, ID 管理の負担を軽減した方式が提案・実装されている [5]. この方式では柔軟な相互認証が可能であり ID 管理の問題は解決しているものの, 二者間の通信を想定しているため複数の機器間での鍵共有ができない. 本研究では, 属性の論理式を満たす複数の機器間で鍵共有が可能となる属性ベース認証付グループ鍵共有方式 [6] を IoT を想定した環境で実装する. そしてその性能を Raspberry Pi で評価し, 測定結果から実装システムが実用的であることを確認する.

[1] 広島大学
Hiroshima University, Higashihiroshima, Hiroshima 739–8527, Japan
[a)] m175758@hiroshima-u.ac.jp
[b)] t-nakanishi@hiroshima-u.ac.jp

2. 従来技術

2.1 属性ベース暗号の概要

属性ベース暗号 (ABE : Attribute-Based Encryption) は ID ベース暗号を拡張したものであり, CP(Ciphertext Poclicy)-ABE と呼ばれるタイプでは, 復号者を属性の条件によって指定できる [4](図 1 参照). まず, ID ベース暗号と同様に鍵サーバが存在し, マスタ秘密鍵と公開鍵を生成し, 公開鍵をユーザに配布する. 次にユーザは自身の複数の属性に対応する秘密鍵を鍵サーバから受け取る. 暗号化を行う際は, 属性のポリシーで暗号化を行う. 属性のポリシーとは AND や OR などの論理演算を使用した属性の論理式である. 復号する場合は, ユーザの属性に対応する秘密鍵を用いて行う. このとき, ユーザの属性が復号ポリシーを満たす場合のみ復号可能となる.

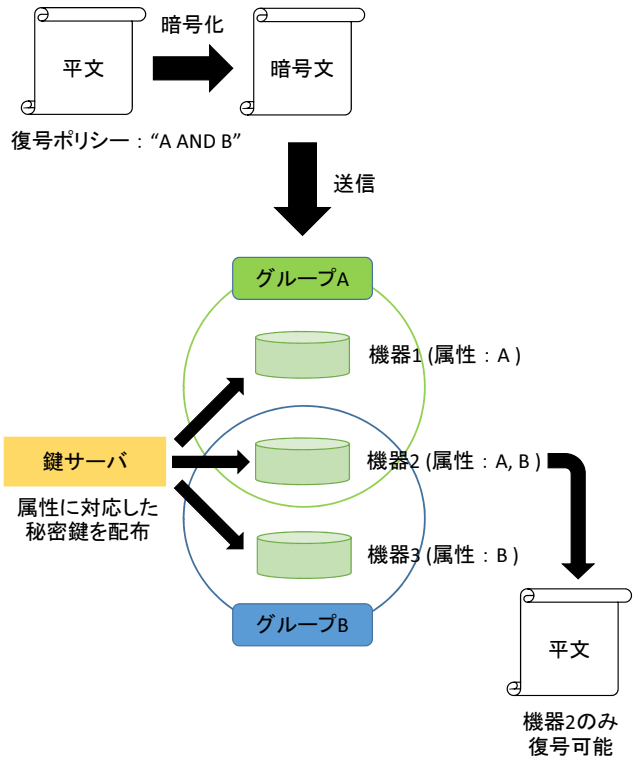

図 1　属性ベース暗号の概要

例えばグループ A に所属する機器 1, グループ B に所属する機器 3, どちらのグループにも所属する機器 2 があり, 通信を復号ポリシー (復号条件) で暗号化しているとする. このとき復号ポリシーを”A”とすると, 機器 1 と機器 2 からの通信のみ復号できるようになる. また, 復号ポリシーは OR や AND などの論理演算を使用することができるため, 復号ポリシーを”A AND B”とすると, 機器 2 のみ通信を復号でき, 復号ポリシーを”A OR B”のようにすると, 機器 1, 機器 2, 機器 3 の全ての通信を復号することができる. このように, 各通信時に動的にかつ柔軟に復号可能な機器を複数指定できることが属性ベース暗号の利点である.

2.2　属性ベース暗号を用いた IoT 向け相互認証

近年, IoT を利用したサービスが普及している. IoT システムでは, 機器とサーバ間で双方向の通信モデルとなっているため, 機器とサーバが相互に正当な相手かどうかを検証する相互認証が必須となる. IoT システムにおける相互認証技術の一つに TLS が挙げられるが, TLS は公開鍵証明書を用いるため, システムにおける機器数が多い IoT 環境では証明書利用による処理・通信量の増大や管理の複雑性が課題となる. そこで TLS の新たな方式として, ID を公開鍵として利用する ID ベース暗号を用いた TLS が提案されている [2]. ID ベース暗号では公開鍵証明書が不要であるため, 証明書の検証や送受信の処理を省くことができる. しかし, ID ベース暗号では機器とそれぞれに ID が付与され, その個別の ID を用いて認証を行うため, 機器数が

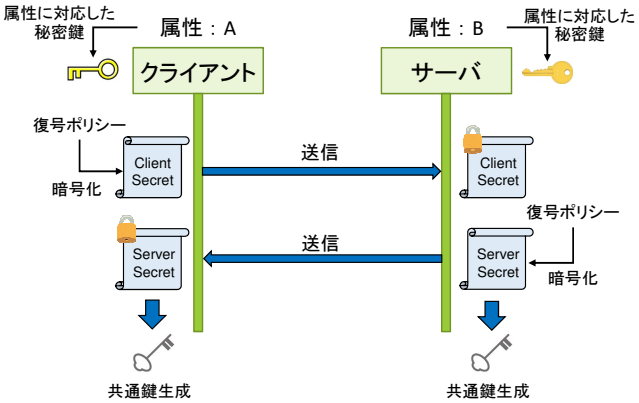

図 2　属性ベース暗号を用いた相互認証・鍵共有プロトコル

非常に多くなる IoT 機器では ID 管理に負担を要する. そこで復号者を属性の条件によってまとめて指定可能な暗号方式である属性ベース暗号を用いることにより, ID 管理の負担を軽減した方式が提案されている [5].

この属性ベース暗号を用いた相互認証 (図 2 参照) では, 事前にサーバ及びクライアントの機器に属性が付与され, 鍵サーバから属性に対応する秘密鍵が配布されている. サーバとクライアントは次のように各々共通鍵を交換する. まずサーバは復号ポリシーを指定し, 自身の秘密情報を暗号化してクライアントへ送信する. クライアントはサーバから暗号文を受信後, 属性に紐づく秘密鍵で復号する. そしてサーバと同様にクライアントも復号ポリシーを指定し, クライアント自身の秘密情報を暗号化してサーバへ送信する. サーバはクライアントと同様に暗号文を受信後, 属性に紐づいた秘密鍵で復号する. 最後にサーバとクラインアントはそれぞれの秘密情報とこれまでやりとりした情報から共通鍵を作成し, 共通鍵の交換が完了する. 共通鍵の交換が完了後, サーバとクライアント間での通信は交換した共通鍵をもとに暗号化通信へ移行する. この方式では各通信時に動的かつ柔軟に復号可能な機器を指定できるメリットがあるが, 二者間での通信を想定しているため複数者間での鍵共有ができないという問題点がある.

2.3　属性ベース認証付鍵交換

属性ベース暗号の枠組みにおいて, 属性ベース認証付鍵交換 (AB-AKE:Attribute-Based Authenticated Key Exchange) が提案されている [6]. 前提として, ある特定の属性ポリシーを満たす属性を持つユーザのグループを想定し, 各ユーザは属性ベース暗号と同様に鍵サーバから各々が持つ属性に応じた秘密鍵が発行される. そしてアクセスポリシーを満たす属性を持っているユーザのみセッションキーを計算できグループ間での鍵交換が完了する. 安全性においては, 属性ベース暗号における結託耐性の性質が AB-AKE に組み込まれている.

2.3.1 EP-AB-KEM の概要

一般的な 1 ラウンドの AB-AKE プロトコルは, EP-AB-KEM(Encapsulation Policy-AB-Key Encapsulation Mechanism）と呼ばれる属性ベースの鍵カプセル暗号化メカニズムに基づいており, ランダムオラクルにおける IND-CCA ベースの安全性が証明されている. EP-AB-KEM では属性と秘密鍵が関連付けられ, アクセスポリシーに基づいて共通鍵がカプセル暗号化される. この方式は [4] の CP-ABE に基づいて構築されている. 以下に EP-AB-KEM の各アルゴリズムを示す.

セットアップ $Setup(k)$

セットアップ $Setup(k)$ はセキュリティパラメータ k を入力とし, 公開パラメータ PK とマスター鍵 MK を出力する. 以下に詳細を示す.

集合 $\mathbb{G}_0, \mathbb{G}_1$ から双線形写像 $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ を定義し, g を $\mathbb{G}_0$ の生成元, $\alpha, \beta_1, \beta_2 \in \mathbb{Z}_p$ とする. ただし $\beta_1 \neq \beta_2, \beta_1 \neq 0, \beta_2 \neq 0$ である. 公開鍵 PK を次のように定義する.

$$PK = (\mathbb{G}_0, \mathbb{G}_1, e, g, h_1 = g^{\beta_1}, f_1 = g^{1/\beta_1}, h_2 = g^{\beta_2}, f_2 = g^{1/\beta_2}, e(g,g)^{\alpha})$$

マスター鍵 MK は $(\beta_1, \beta_2, g^{\alpha})$ である.

カプセル暗号化 $Encapsulation(PK, T)$

カプセル暗号化 $Encapsulation(PK, T)$ は公開パラメータ PK とアクセスポリシー T を入力とし, カプセル暗号文 C と共通鍵 K を出力する. 属性ベース暗号と同様に, アクセスポリシー T を満たす属性を持つユーザだけがカプセル暗号化 C から共通鍵 K を復号できる. 以下に詳細を示す.

ワンタイム署名の鍵ペア (sk, vk) を生成する. 乱数 $s \in \mathbb{Z}_p$ とし, カプセル暗号化のアルゴリズム手順は次のようになる. ここで T' は (T AND vk) となり, T' のアクセス木のノード x に対する多項式を q_x とする. また Y を T のアクセス木における葉ノードの集合とする. そして, 以下を行う.

1. $K = e(g,g)^{\alpha s}$
2. $C_1 = h_1^s$
3. $\forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}$
4. $C_{vk} = h_2^{q_{vk}(0)}, C'_{vk} = H(vk)^{q_{vk}(0)}$
5. $C = (T', C_1, C_y, C'_y, C_{vk}, C'_{vk}), \forall y \in Y$

最後に署名 $\sigma = Sig_{sk}(C)$ としてカプセル暗号文を $C = (C, vk, \sigma)$ とする.

鍵生成 $KeyGen(MK, PK, S)$

鍵生成 $KeyGen(MK, PK, S)$ はマスター鍵 MK と公開パラメータ PK とユーザの属性集合 S を入力とし, ユーザの秘密鍵 SK を出力する. 以下に詳細を示す.

$j \in S, r, r_{vk} \in \mathbb{Z}_p, r_j \in \mathbb{Z}_p$ として秘密鍵を次のように定義する.

$$SK = (D = g^{(\alpha+r)/\beta_1}, E = g^{r/\beta_2}, \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j})$$

カプセル復号 $Decapsulation(SK, PK, C)$

カプセル復号 $Decapsulation(SK, PK, C)$ は公開パラメータ PK とカプセル暗号文 C とユーザの秘密鍵 SK を入力とし, ユーザの属性集合 S がアクセスポリシー T を満たせば共通鍵 K を出力する. 以下に詳細を示す.

ユーザはカプセル暗号文 C を受信後, 検証鍵 vk を用いて C に含まれる署名 σ を検証する. 検証に成功すると, 次のように処理する.

$$\begin{aligned} F_{vk} &= \frac{e(C_{vk}, H(vk) \cdot g^{r/\beta_2})}{e(C'_{vk}, h_2)} \\ &= \frac{e(C_{vk}, g^{r/\beta_2}) \cdot e(C_{vk}, H(vk))}{e(C'_{vk}, h_2)} \\ &= \frac{e(h_2^{q_{vk}(0)}, g^{r/\beta_2}) \cdot e(h_2^{q_{vk}(0)}, H(vk))}{e(H(vk)_2^{q_{vk}(0)}, h_2)} \\ &= e(g^{\beta_2 \cdot q_{vk}(0)}, g^{r/\beta_2}) = e(g,g)^{r q_{vk}(0)} \end{aligned}$$

次にカプセル暗号文 C と秘密鍵 SK およびアクセスポリシー T の木の各ノード x を入力とし, $DecryptNode(C, SK, x)$ が実行される. x が葉ノードのときは以下のようになる. ここで $i = att(x)$(x の属性のインデックス) とする.

$DecryptNode(C, SK, x) = \frac{e(D_i, C_x)}{e(D'_i, C'_x)} = \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} = e(g,g)^{r q_x(0)}$

x が内部ノードのときは, すべての子ノード x に対して $DecryptNode(C, SK, z)$ が実行され出力は F_z に格納される. ここで x の子ノードの集合を S_x とするとカプセル復号アルゴリズムは以下のように実行される. ただし $i = index(z)$, $S'_x = \{index(z) : z \in S_x\}$, Lagrange 係数 $\Delta_{i,S'_x} = \prod_{j \in S'_x,\ j \neq i} \frac{x-j}{i-j}$ とする.

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r \cdot q_z(0)})^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r \cdot q_x(i) \cdot \Delta_{i,S'_x}(0)} \\ &= (e(g,g)^{r \cdot q_x(0)} \end{aligned}$$

最後に, アクセスポリシー木 T のルートである R において $DecryptNode$ アルゴリズムを実行する. ユーザの属性集合 S がアクセスポリシー T を満たす場合, $F_R = DecryptNode(C, SK, R) = e(g,g)^{r \cdot q_R(0)}$ となる. また F_R と F_{vk} から $F_{R'}$ を計算すると次のようになる.

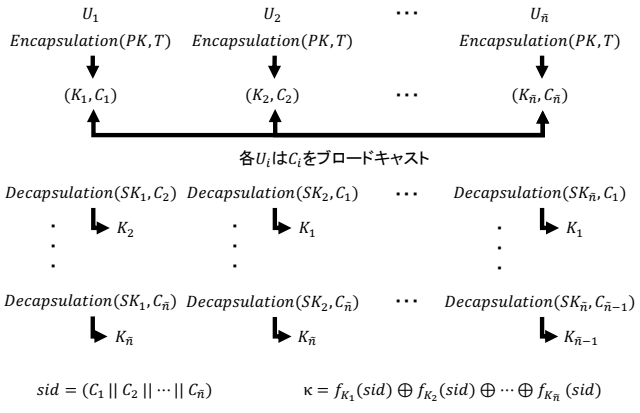

図 3　属性ベース認証付鍵共有の概要

$$
\begin{aligned}
F_{R'} &= \prod_{x \in \{R,vk\}} F_x^{\Delta_{index(x),\{R,vk\}}} \\
&= e(g,g)^{r \cdot q_{R'}(0)} \\
&= e(g,g)^{rs}
\end{aligned}
$$

ここで $A = e(g,g)^{rs}$ とすると, 共通鍵を復号することができる.

$$
\begin{aligned}
\frac{e(C_1,D)}{A} &= \frac{e(h_1^s, g^{(\alpha+r)/\beta_1})}{e(g,g)^{rs}} \\
&= \frac{e(g,g)^{s(\alpha+r)})}{e(g,g)^{rs}} \\
&= e(g,g)^{\alpha s} = K
\end{aligned}
$$

2.3.2　EP-AB-KEM を用いた属性ベース認証付鍵交換のプロトコル

以下に EP-AB-KEM を用いた属性ベース認証鍵交換のプロトコル [6] を示す (図 3 参照).

1. カプセル暗号化の計算

PK をマスター公開鍵, T をアクセスポリシーを表すアクセス木とする. 各ユーザ U_i は入力 (PK,T) において EP-AB-KEM を実行しカプセル化 (Encapsulation) を行う. その結果, 共通鍵 K_i とカプセル C_i のペア (K_i,C_i) を得る.

$$(K_i,C_i) \longleftarrow Encapsulation(PK,T)$$

2. カプセル化のブロードキャスト

各ユーザ U_i は生成されたカプセル C_i をブロードキャストする.

$$U_i \longrightarrow * : C$$

3. 鍵の計算

各ユーザ U_i は受信したカプセル C_j に対し自身の秘密鍵 SK_i を用いてカプセル解除 (Decapsulation) を実行し共通鍵 K_i を得る. ただし $j \neq i$.

$$K_j \longleftarrow Decapsulation(SK_i,C_j), j \neq i$$

各ユーザ U_i は送受信したすべてのメッセージを連結し, それをセッション ID として計算する. すなわちセッション ID は $sid = (C_1 \| \cdots \| C_{\tilde{n}})$ となる. $\tilde{n}$ はプロトコルの参加数である.

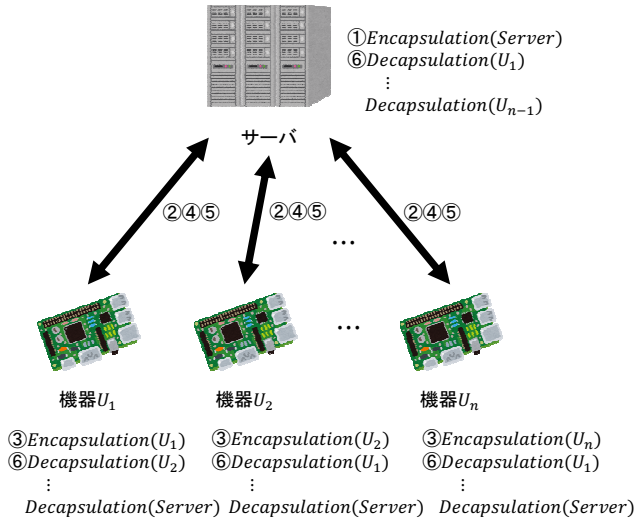

図 4　IoT 向け属性ベースグループ鍵共有プロトコル

セッション鍵 κ は以下のように計算される.

$$\kappa = f_{K_1}(sid) \oplus f_{K_2}(sid) \oplus \cdots \oplus f_{K_{\tilde{n}}}(sid)$$

ただし f は擬似ランダム関数である.

3.　IoT 向け属性ベースグループ鍵共有プロトコル

従来方式 [5] では二者間の通信を想定しているため複数者間での鍵共有ができないという課題があった. その課題を解決するために, 本研究では属性ベース認証付鍵交換 [6] を用いて, IoT を想定したグループ鍵共有プロトコルを実装する. ここでは, 1 台のサーバ (クラウドサーバもしくはエッジサーバ) と n 台の IoT 機器を想定する. 以下にプロトコルの手順を示す (図 4 参照).

① サーバはアクセスポリシーを指定し, マスター公開鍵 PK とアクセスポリシー T を用いてカプセル暗号化する.

② サーバは生成したカプセル暗号文 C_{Server} をブロードキャストし, すべての機器 $U_1 \sim U_n$ に送信する.

③ 各機器 $U_1 \sim U_n$ はサーバからのカプセル暗号文 C_{Server} を受信した後, サーバと同様にカプセル暗号化を実行する.

④ 各機器 $U_1 \sim U_n$ はそれぞれ生成したカプセル暗号文 $C_{U_1} \sim C_{U_n}$ をサーバに送信する.

⑤ サーバは各機器 $U_1 \sim U_n$ にその機器以外が生成したカプセル暗号文すべてを送信する.

⑥ サーバおよび各機器は受信したすべてのカプセル暗号文に対してカプセル復号処理を行う.

⑦ サーバおよび各機器はこれまでやりとりした情報からセッション ID を計算し, セッション鍵 κ を入手する.

4.　実装結果

本研究では, 前節の IoT 向け属性ベースグループ鍵共有プロトコルをサーバを想定した汎用 PC と IoT 機器を想定した超小型 PC である Raspberry Pi に実装した. 本節で

は, これらの機器間で必要なカプセル暗号化・復号処理や鍵計算および通信にかかる時間などを計測し評価する.

4.1 実装環境

使用した PC および Raspberry Pi の環境を表 1 に示す. 実装では, CP-ABE 方式 [4] の各処理のプログラムが提供されている CPABE(Ciphertext-Policy Attribute- Based Encryption) ライブラリ [7] に対して, そのベースとなっているペアリングを含む楕円曲線演算の PBC(Pairing-Based Cryptography) ライブラリ [8] を用いて拡張した. PBC ライブラリでは, 多倍長演算には GMP ライブラリが使用されている.

表 1 計測に使用した機器の環境

	PC
CPU	Intel Core i5-6400(2.70GHz × 4)
OS	ubuntu 14.04 LTS (32 ビット)
メモリ	7.8 GiB
	Raspberry Pi
CPU	ARM Cortex-A53(1.2GHz)
OS	GNU/Linux
メモリ	1 GB
使用言語	C 言語
ライブラリ	cpabe-0.11(属性ベース暗号ライブラリ)
	pbc-0.5.14(ペアリングライブラリ)
	gmp-6.1.1(多倍長演算ライブラリ)

4.2 評価方法と結果

測定実験では IoT 機器数を 2 とし, 通信環境として広島大学キャンパス情報ネットワーク HINET および無線 LAN(IEEE 802.11bgn) を使用している.

評価実験では, サーバと機器の属性数をともに 3 個とし, サーバがカプセル化を開始してからセッション鍵を生成するまでの時間を 1 ラウンドとして計測した. またサーバや各機器がカプセル暗号化やカプセル復号に要する時間やセッション鍵生成および通信にかかる時間もそれぞれ計測した. ここで表 2 に鍵交換の 1 ラウンドに必要な時間を, 図 5 に各処理時間の占める割合を示す.

計測結果から属性ベース暗号を用いたグループ間鍵共有に必要な 1 ラウンドの時間は約 372ms であり, 実用的な時間であることがわかる. サーバと機器 U_1, U_2 ではサーバの方が処理性能が高いので, カプセル暗号化やカプセル復号などの処理においてサーバの方が実行時間が短い. また本実装では③,⑥,⑦の処理は各サーバや機器が各々並列に実行するため, サーバが機器よりも先にカプセル復号およびセッション鍵を取得することができるが, セッション鍵を用いた通信をするためには各機器のカプセル復号およびセッション鍵の処理時間を待つ必要がある. ただし③の暗号処理や⑥,⑦の復号処理は最大でも一番遅い機器の処理

表 2 鍵交換プロトコルでの各処理時間

各処理	時間 [ms]
① サーバのカプセル暗号化 $Encapsulation(Server)$	**51.5**
② サーバのブロードキャスト	**2.61**
③ 各機器のカプセル暗号化	**146.1**
U_1 のカプセル暗号化 $Encapsulation(U_1)$	144.5
U_2 のカプセル暗号化 $Encapsulation(U_2)$	146.1
④ U_1 からサーバへの通信時間	**36.7**
④ U_2 からサーバへの通信時間	**36.8**
⑤ サーバから U_1 への通信時間	**2.55**
⑤ サーバから U_2 への通信時間	**2.60**
⑥ カプセル復号処理 (1 回目)	**46.6**
サーバのカプセル復号 $Decapsulation(U_1)$	12.2
U_1 のカプセル復号 $Decapsulation(U_2)$	46.6
U_2 のカプセル復号 $Decapsulation(U_1)$	46.5
⑦ カプセル復号処理 (2 回目)	**46.8**
サーバのカプセル復号 $Decapsulation(U_2)$	12.1
U_1 のカプセル復号 $Decapsulation(Server)$	46.7
U_2 のカプセル復号 $Decapsulation(Server)$	46.8
1 ラウンドの時間	**372.3**

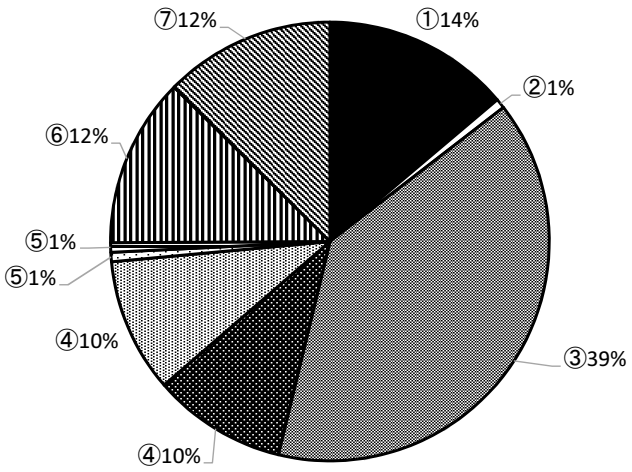

図 5 鍵交換プロトコルでの各処理時間の割合

時間だけ待てば良いため, 機器数が増えても待機時間が増えることはない. 一方, サーバ・各機器での復号の回数は n となるため, グループの機器数に比例して 1 ラウンドの時間は増加すると考えられる.

図 5 からは 1 ラウンドに必要な時間のうち最も多くの割合を占めている処理はカプセル暗号化であり, 次にカプセル復号となることが分かる. これは従来方式の相互認証 [5] においても類似した結果となっており, 属性ベース暗号を用いたペアリングによる暗号化や復号の処理が鍵交換の大半を占めている.

5. まとめ

本研究では, IoT 環境における属性ベース暗号を用いたグループ間鍵共有プロトコルの実装を行った. また PC お

よび Raspberry Pi 上で実装し, サーバ 1 台, IoT 機器 2 台の環境で計測したところ, 測定結果からグループ間での鍵共有に要する 1 ラウンドの時間が約 372ms となり, 本プロトコルの実用性を検証することができた.

今回の提案方式の計測ではサーバの数を 1, 機器数を 2 とした 3 者間でのグループ鍵共有として実験したが, 機器数を増やした際に通信方法が複雑になる可能性があるため, 効率良く鍵交換を行えるような新たな通信手順を検討している.

謝辞 本研究の一部は，JSPS 科研費 (JP16H01723) の助成を受けて行われている.

参考文献

[1] J.Viega, Matt Messier, Pravir Chandra, 『OpenSSL — 暗号・PKI・SSL/TLS ライブラリの詳細 — 』, オーム社.
[2] 酒見由美, 武仲正彦, 金岡晃, 「ID ベース暗号による IoT 向け相互認証方式の提案」, The 32nd Symposium on Cryptography and Information Security, SCIS 2015.
[3] 酒見由美, 伊豆哲也, 武仲正彦, 金岡晃, 「事前共有鍵に基づく TLS の ID ベース暗号による拡張」, 信学技報, ISEC2013-43, IEICE, 2013.
[4] J.Bethencourt, A.Sahai, B.Waters, "Ciphertext-Policy Attribute-Based Encryption", IEEE Symposium on Security and Privacy, pp.321 334, 2007.
[5] 椿雄介, 中西透, 「属性ベース暗号を用いた IoT 向け相互認証の実装」, 信学技報, ISEC2017-70, IEICE, 2017.
[6] M.Choudary Gorantla,Colin Boyd,Juan Manuel Gonzalez Nieto, "Attribute-Based Authenticated Key Exchange", ACISP, pp.300 317, 2010.
[7] J.Bethencourt, A.Sahai, B.Waters, 「Advanced Crypto Software Collection」, http://acsc.cs.utexas.edu/cpabe/, 2018/11/12 アクセス.
[8] B.Lynn「PBC Library — The Pairing-Based Cryptography —」, https://crypto.stanford.edu/pbc/, 2018/11/12 アクセス.